## Session 7

**Enhanced Ball Animation using PyGame**

## This is what we have done so far in the Ball animation

**CURRENT PROCESS:**

The balls move out of the screen as if nothing is stopping them from doing that.

import pygame  
from random import randint  
  
pygame.init()  
  
screen = pygame.display.set\_mode([500, 500]) *# creates a screen with the said size*  
  
clock = pygame.time.Clock()  
  
  
*# Creating the Base/Parent circle class*  
**class** Circle():  
 **def** \_\_init\_\_(self):  
 self.x = randint(0,500)  
 self.y = randint(0,500)  
 self.r = randint(10,50)  
 self.color = (randint(0,255),randint(0,255),randint(0,255))  
 self.x\_speed = randint(-2,2)  
 self.y\_speed = randint(-2,2)  
   
 **def** move(self):  
 self.x = self.x+self.x\_speed  
 self.y = self.y+self.y\_speed  
   
 *# Inheriting from the parent circle class and creating the child class for FasCtCircle*  
**class** FastCircle(Circle):  
 **def** \_\_init\_\_(self):  
 super().\_\_init\_\_()  
   
 *# redefinig the move() method*  
 **def** move(self):  
 self.x = self.x + (self.x\_speed\*2)  
 self.y = self.y + (self.y\_speed\*2)  
   
 *# drawing circles with thin edges*  
 **def** draw(self):  
 pygame.draw.circle(screen, self.color, (self.x, self.y), self.r,2)  
   
   
*# Inheriting from the parent circle class and creating the child class for SlowCircle*  
**class** SlowCircle(Circle):  
 **def** \_\_init\_\_(self):  
 super().\_\_init\_\_()  
   
 *# There is no need to define the move method as we can use the one from the parent class*  
   
 *# drawing circles with thin edges*  
 **def** draw(self):  
 pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)  
  
*# List to store all the circles*   
cir = []  
  
*# Creating FastCircles and adding them to the list*  
**for** i **in** range(5):  
 cir.append(FastCircle())  
  
*# Creating SlowCircles and adding them to the list*  
**for** i **in** range(5):  
 cir.append(SlowCircle())  
  
  
run = True  
**while** run:  
 **for** event **in** pygame. event.get():  
 **if** event.type == pygame.QUIT:  
 run = False  
   
 screen.fill((255,255,255))  
  
 *# Creating individual circle objects using the draw method*  
 **for** i **in** range(10):  
 cir[i].draw()  
   
 *# Moving the circle objects*  
 **for** i **in** range(10):  
 cir[i].move()  
   
   
 pygame. display.flip()  
   
 clock.tick(30)  
   
   
pygame.quit()

**NEXT STEP:**

The balls should not move out of the screen, so we introduce the bounce animation after hitting the wall.

### So let's try building on the ball animation and create a wallpaper out of it

1. The balls should bounce off the walls after hitting it
2. The color of the ball should change after it hits the wall

**Detecting the ball hitting the wall can be done using a simple formula** **Every wall on the screen will have a separate formula for detecting a ball collision with it**

**The below formula detects the collision of the ball with the right-hand side wall** x+radius > 500

If the above condition becomes true it means the ball has collided with the right-hand side wall and the ball should reverse its direction. This reversal of direction has to happen only in the horizontal direction ie we resverse the x speed of the ball. As the ball can retain its speed in the vertical direction.

Just like the above condition for the right-hand side, we can come up with a condition for all the walls on the screen.

**The code below shows how to detect if the ball has collided with any wall on the screen.**

This has to be added as a method to the circle class as all the balls are required to detect if it has hit the wall and reverse the direction.

### The balls should bounce off the walls after hitting it

**def** wall\_collide(self):  
 *# Detecting collision with the right wall*  
 **if** self.x+self.r > 500:  
 *# reversing the speed which also reverses the direction of the ball movement*  
 self.x\_speed = self.x\_speed \* -1  
   
 *# Detecting collision with the left wall*  
 **if** self. x-self.r < 0:  
 *# reversing the speed which also reverses the direction of the ball movement*  
 self.x\_speed = self.x\_speed \* -1  
   
 *# Detecting collision with the bottom wall*  
 **if** self.y+self.r > 500:  
 *# reversing the speed which also reverses the direction of the ball movement*  
 self.y\_speed = self.y\_speed \* -1  
   
 *# Detecting collision with the top wall*  
 **if** self. y-self.r < 0:  
 *# reversing the speed which also reverses the direction of the ball movement*  
 self.y\_speed = self.y\_speed \* -1

### Putting it all together

import pygame  
from random import randint  
  
pygame.init()  
  
screen = pygame.display.set\_mode([500, 500]) *# creates a screen with the said size*  
  
clock = pygame.time.Clock()  
  
  
*# Creating the Base/Parent circle class*  
**class** Circle():  
 **def** \_\_init\_\_(self):  
 self.x = randint(0,500)  
 self.y = randint(0,500)  
 self.r = randint(10,50)  
 self.color = (randint(0,255),randint(0,255),randint(0,255))  
 self.x\_speed = randint(-2,2)  
 self.y\_speed = randint(-2,2)  
   
 **def** move(self):  
 self.x = self.x+self.x\_speed  
 self.y = self.y+self.y\_speed  
   
 **def** wall\_collide(self):  
 **if** self.x+self.r > 500:  
 self.x\_speed = self.x\_speed \* -1  
 **if** self. x-self.r < 0:  
 self.x\_speed = self.x\_speed \* -1  
 **if** self.y+self.r > 500:  
 self.y\_speed = self.y\_speed \* -1  
 **if** self. y-self.r < 0:  
 self.y\_speed = self.y\_speed \* -1

*# Inheriting from the parent circle class and creating the child class for FasCtCircle*  
**class** FastCircle(Circle):  
 **def** \_\_init\_\_(self):  
 super().\_\_init\_\_()  
   
 *# redefinig the move() method*  
 **def** move(self):  
 self.x = self.x + (self.x\_speed\*2)  
 self.y = self.y + (self.y\_speed\*2)  
   
 *# drawing circles with thin edges*  
 **def** draw(self):  
 pygame.draw.circle(screen, self.color, (self.x, self.y), self.r,2)  
   
   
*# Inheriting from the parent circle class and creating the child class for SlowCircle*  
**class** SlowCircle(Circle):  
 **def** \_\_init\_\_(self):  
 super().\_\_init\_\_()  
   
 *# There is no need to define the move method as we can use the one from the parent class*  
   
 *# drawing circles with thin edges*  
 **def** draw(self):  
 pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)  
  
*# List to store all the circles*   
cir = []  
  
*# Creating FastCircles and adding them to the list*  
**for** i **in** range(50):  
 cir.append(FastCircle())  
  
*# Creating SlowCircles and adding them to the list*  
**for** i **in** range(50):  
 cir.append(SlowCircle())  
  
  
run = True  
**while** run:  
 **for** event **in** pygame. event.get():  
 **if** event.type == pygame.QUIT:  
 run = False  
   
 *#screen.fill((255,255,255))*  
   
   
  
 *# Creating individual circle objects using the draw method*  
 **for** i **in** range(100):  
 cir[i].draw()  
   
 *# Moving the circle objects*  
 **for** i **in** range(100):  
 cir[i].move()  
   
 *# chekcing if the balls are colliding with the wall*  
 **for** i **in** range(100):  
 cir[i].wall\_collide()  
   
   
   
 pygame. display.flip()  
   
 clock.tick(30)  
   
   
pygame.quit()

### Try changing the ball's color as soon as it hits the wall. Choose any random color

**Solution:**

import pygame  
from random import randint  
  
pygame.init()  
  
screen = pygame.display.set\_mode([500, 500]) *# creates a screen with the said size*  
  
clock = pygame.time.Clock()  
  
  
*# Creating the Base/Parent circle class*  
**class** Circle():  
 **def** \_\_init\_\_(self):  
 self.x = randint(0,500)  
 self.y = randint(0,500)  
 self.r = randint(10,50)  
 self.color = (randint(0,255),randint(0,255),randint(0,255))  
 self.x\_speed = randint(-2,2)  
 self.y\_speed = randint(-2,2)  
   
 **def** move(self):  
 self.x = self.x+self.x\_speed  
 self.y = self.y+self.y\_speed  
   
 **def** wall\_collide(self):  
 **if** self.x+self.r > 500:  
 self.x\_speed = self.x\_speed \* -1  
 self.color = (randint(0,255),randint(0,255),randint(0,255))  
 **if** self. x-self.r < 0:  
 self.x\_speed = self.x\_speed \* -1  
 self.color = (randint(0,255),randint(0,255),randint(0,255))  
 **if** self.y+self.r > 500:  
 self.y\_speed = self.y\_speed \* -1  
 self.color = (randint(0,255),randint(0,255),randint(0,255))   
 **if** self. y-self.r < 0:  
 self.y\_speed = self.y\_speed \* -1  
 self.color = (randint(0,255),randint(0,255),randint(0,255))  
   
   
   
*# Inheriting from the parent circle class and creating the child class for FasCtCircle*  
**class** FastCircle(Circle):  
 **def** \_\_init\_\_(self):  
 super().\_\_init\_\_()  
   
 *# redefinig the move() method*  
 **def** move(self):  
 self.x = self.x + (self.x\_speed\*2)  
 self.y = self.y + (self.y\_speed\*2)  
   
 *# drawing circles with thin edges*  
 **def** draw(self):  
 pygame.draw.circle(screen, self.color, (self.x, self.y), self.r,2)  
   
   
*# Inheriting from the parent circle class and creating the child class for SlowCircle*  
**class** SlowCircle(Circle):  
 **def** \_\_init\_\_(self):  
 super().\_\_init\_\_()  
   
 *# There is no need to define the move method as we can use the one from the parent class*  
   
 *# drawing circles with thin edges*  
 **def** draw(self):  
 pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)  
  
*# List to store all the circles*   
cir = []  
  
*# Creating FastCircles and adding them to the list*  
**for** i **in** range(5):  
 cir.append(FastCircle())  
  
*# Creating SlowCircles and adding them to the list*  
**for** i **in** range(5):  
 cir.append(SlowCircle())  
  
  
run = True  
**while** run:  
 **for** event **in** pygame. event.get():  
 **if** event.type == pygame.QUIT:  
 run = False  
   
 *#screen.fill((255,255,255))*  
   
   
  
 *# Creating individual circle objects using the draw method*  
 **for** i **in** range(10):  
 cir[i].draw()  
   
 *# Moving the circle objects*  
 **for** i **in** range(10):  
 cir[i].move()  
   
 *# chekcing if the balls are colliding with the wall*  
 **for** i **in** range(10):  
 cir[i].wall\_collide()  
   
   
   
 pygame. display.flip()  
   
 clock.tick(30)  
   
   
pygame.quit()

## Task 1 :

### Let's try fixing a few issues with the wallpaper animation

1. Fix the issue where the balls spawn too close to the edge and get stuck over there. (**Hint: Change the initial x and y coordinates of the ball**)
2. Fix the issue in which the ball remains stationary on the screen. (**Hint: try using random.choices()**)

**REVISION**

* Concept of: OOP, Class, Object, Method, Attributes, Inheritance, polymorphism
* All programs in sequence
* Animation (effect)
* Wallpaper making

**HOMEWORK**

1. Try making the balls collide with each other
2. Try to make code for bouncing rectangle